Concept learning from examples

Flach, P.A.; Veelenturf, L.P.J.

Publication date:
1989

Document Version
Publisher's PDF, also known as Version of record

[redacted]

ITK

RESEARCH
REPORT

INSTITUTE FOR LANGUAGE TECHNOLOGY AND ARTIFICIAL INTELLIGENCE

INSTITUUT VOOR TAAL- EN KENNISTECHNOLOGIE

ITK Research Report No. 2

January 1989

Concept Learning from Examples: Theoretical Foundations

Peter A. Flach

Leo P J. Veelenturf

Institute for Language Technology and Artificial Intelligence ITK

Tilburg University, Tilburg, the Netherlands

Concept Learning from Examples: Theoretical Foundations

ABSTRACT

A mathematical model of concept learning from examples is presented. This model bears a set-theoretical nature, and serves as a semantics for a symbolic description language. This semantics naturally leads to an algebraic theory of learning, in which properties of specific concept expressions (e.g., conjunctive expressions) can be proved. Important feature of the model is an incorporation of the notion of background knowledge. The model describes also how learning systems can deal with examples that are not completely specified by the teacher (incomplete knowledge). It is indicated how the proof-theoretical framework of Predicate Logic could be used within a learning system, and how an environment for implementing learning algorithms (a learning shell) could be developed on the basis of the algebraic theory.

Contents

Concept Learning from Examples: Theoretical Foundations

Peter A. Flach

Tilburg University
Institute for Language Technology and Artificial Intelligence
Tilburg, the Netherlands

Leo P J. Veelenturf

Twente University
Department of Electrical Engineering
Enschede, the Netherlands

1. Introduction

The ability to learn from past experiences seems to be present, in one form or another, in most animals. Human beings, however, have the distinguished capability of making the newly induced knowledge explicit. Indeed, this ability of learning by explicit knowledge acquisition is one of the main features of human intelligence. For this reason, it is a major topic in the cognitive sciences.

In the field of Artificial Intelligence, being a cognitive science, the topic related to human learning behaviour is called *machine learning*. The aim of this subfield of research is to build (or describe formally) machines that simulate aspects of human learning behaviour. It is generally acknowledged that machine learning is a central topic in Artificial Intelligence research. For instance, the known bottleneck in current expert systems is the process of knowledge acquisition: that is, transforming the knowledge of a human expert in a certain field into a consistent and complete set of knowledge-based rules. Other possible applications of learning machines are in systems for automated programming, and in robotics.

Learning machines manifest themselves in many forms, and several attempts have been made to develop meaningful classifications of learning machines [1, 2, 3]. Still, leading researchers admit that the subfield of machine learning lacks a sound foundational theory and a basic vocabulary [4]. This makes it hard to compare work of different researchers, and to distinguish between significant new results and older achievements.

This applies equally to the type of machine learning that is generally referred to as 'the oldest and best understood problem in Artificial Intelligence' [5]: **concept learning from examples**, which is the subject of this paper. The present authors agree with this qualification if interpreted as a statement about the characteristics of concept learning from examples: a process which takes as input descriptions of positive and negative examples, and yields as output a description of a concept, which is consistent with the input. Yet, we disagree with the claim that is suggested by the quoted statement, namely that the meaning of keywords like 'description' and 'consistent' is well understood. The nature of concept learning from examples still needs a thorough analysis by means of mathematical methods. It is this formal analysis that is attempted in this paper.

The research, of which some results are described here, originates not from an Artificial Intelligence point of view, but from an Electrical Engineering point of view. At Twente University, research has been conducted on the subject of deriving a fuse table for Programmable Logic Arrays from a partial specification of desired input/output-behaviour, instead of deriving it from given Boolean next state and output functions (which is a complete specification of input/output-behaviour). Also, research was conducted on growing (learning) neuron networks modelled as state diagrams, the process of growth being controlled by stimulus/response-pairs (examples) [6]. As these lines of research progressed, it was felt that their mean denominator was 'learning from examples' and a research project was started on this topic. In order to achieve widely applicable results, attention was directed towards symbolic

learning or concept learning, and a survey was made of present Artificial Intelligence theories on concept learning from examples.

One of the most striking results of this survey was, that most work on machine learning (and, perhaps, on Artificial Intelligence as a whole), is concerned with algorithms and implementations thereof. Very little recent work on theoretical foundations of learning machines was found. This is perhaps best exemplified by the fact, that the starting point for our research was a chapter from a book published in 1969 [7]. Yet, the present authors believe that algorithms can only be justified (and implementations be verified!) if the underlying model is made explicit. This is no less valid for more or less vague subjects like 'human intelligence' or 'learning behaviour'. Formal models make the basic assumptions visible, and reveal fundamental limitations. As Bundy puts it ([8] pp.49-50):

> "The Artificial Intelligence literature abounds with plausible looking formalisms, without a proper semantics. As soon as you depart from the toy examples illustrated in the paper, it becomes impossible to decide how to represent information in the formalism or whether the processes described are reasonable or what these processes are actually doing."

And Turner remarks ([9] p.119):

> "Indeed, often very little attempt is made [by most AI workers] to justify their formalisms from any sort of semantic perspective. My hope is that this state of affairs will be short-lived."

The approach of this paper can be characterised as logical, or rather, Boolean. We stick to Boolean algebra and first order Predicate Logic, not because we feel it is appropriate for developing and implementing learning algorithms, but because no more is needed for the purposes of this paper. The next section is devoted to an informal discussion of the main issues of concept learning from examples, with emphasis on descriptions of concepts and examples. In section 3, a set-theoretical model is introduced, which serves as a semantics of a description language. As has been mentioned, this model is based upon ideas originally proposed by Ranan Banerji. In section 4, a description language is introduced for use in the rest of the paper. In section 5 the task of concept learning from examples is defined. Section 6 describes the set-theoretical model in an algebraic way. Section 7 discusses possible forms of background knowledge. In section 8, it is indicated how Predicate Logic could be used as a description language. As pointed out in section 9, this also enables one to use the proof-theoretical framework of Predicate Logic in a learning system. Alternatively, the algebraic approach of section 6 could be implemented. In section 10, the main conclusions are listed.

The most important conclusions reached in this paper have a mathematical nature and are therefore expressed as FACTS (i.e., theorems without proofs). Nevertheless, all of them can be proved [10]. As usual, the word 'iff' denotes necessary and sufficient conditions ('if and only if').

2. Concept learning from examples

We will start our investigation by defining what is meant by **concept learning from examples**. From a behaviorist point of view, *concept learning* is the process of acquiring the ability of discriminating between members and non-members of the class of objects of a given concept. We will add a cognitive flavour by restricting ourselves to learning processes where such discrimination takes place on the basis of some formula, which is called the *internal representation* of the concept. Typically, such an internal representation will consist of a *descriptional* part (e.g., 'If an object satisfies such and such a description,') and a *computational* part (e.g., 'the probability that it belongs to the concept is p'). In this paper, we will ignore this computational part of the internal representation; the latter will also be called the *(internal) description* of the class of objects of the concept to be learned.

The **process** of concept learning may be looked upon in different ways. For instance, we may view it as a process of subsequent refinement of the internal representation, as searching through a space of possible internal representations, or as the derivation of the internal representation in a (apparently non-deductive) logic calculus. As has been stated in the previous section, we will not dwell on computational (or algorithmic) issues, but focus attention on the descriptional aspects of concept learning (which is, indeed, a prerequisite for devising a learning algorithm).

By concept learning **from examples** we mean learning processes based on examples of the concept to be learned (positive and/or negative). An *example* of a concept is a description of a member (or non-member, in the case of a negative example) of the concept (what is meant by the term 'description' will be explained shortly). In general, we assume that a set of positive examples and a possibly empty set of negative examples is given, although we will frequently refer to them as being supplied by a *teacher*. Also, we speak of the entity that undergoes the learning process as the *learner*; it may be supposed to be a computer, programmed by a learning algorithm. We will feel free to use other words in an anthropomorphic sense.

Sometimes, the teacher provides explicit background-knowledge along with the examples. By *background-knowledge* we mean (formalised) knowledge that enables the learner to relate examples to each other in a non-trivial way. We will return to this issue in section 7.

In order to explain the use of the term 'example' in this paper, consider the following two games. Both of these games require a teacher and a learner (both of them should be humans). The teacher has a certain concept in mind, and the learner tries to discover this 'concept to be learned'. In the first game, called *non-verbal concept learning*, the teacher shows several objects to the learner and classifies each of these objects as either positive or negative, according to whether the object belongs to the concept he has in mind. The learner is allowed to conduct experiments on these objects in order to determine their relevant properties.

The second game is called *verbal concept learning*. Instead of showing objects to the learner, the teacher gives descriptions of objects, while again classifying these descriptions as either positive or negative, according to whether the object so described belongs to the concept he has in mind. The learner has to discover this concept on the basis of these descriptions alone.

Several observations can be made concerning the games just proposed. In both games, the learner's degree of success depends heavily on the objects the teacher chooses. That is, the more these objects are characteristic for the concept to be learned, the better the learner will be able to discover it. This observation applies to concept learning from examples in general.

Now, suppose the two games are played simultaneously, both with the same teacher but with different learners. The teacher has in both games the same concept in mind and chooses the same objects. If we call a presented object in the non-verbal game an *example-object*, and a presented description in the verbal game an *example-description*, then the (rather trivial) statement can be made that each example-object satisfies the corresponding example-description. However, this statement can **not** be reversed; it is, in general, not true that any object that satisfies a, for instance positive, example-description can serve as a positive example-object. It may be that some of the characteristic properties of the example-object have been left out of the example-description; a positive example-description and a negative one may even be the same! In the verbal game, the learner will never be able to decide whether all characteristic features of an example-object are contained in its example-description.

In the verbal game, an example-description represents a set of objects rather than a single object, some of which may belong to the concept to be learned and some of which may not. From this it follows that verbal concept learning is at least as difficult as non-verbal concept learning. It is clear, however, that in **machine learning** the verbal game is the interesting case. The learning machine has to be able to deal with **incomplete knowledge**. In the next section a set-theoretical model for (verbal) concept learning from examples will be developed.

3. A set-theoretical model

The set-theoretical framework for concept learning from examples as described in this section has originally been proposed by R.B. Banerji [7]. Banerji uses this framework for describing patterns, with emphasis on what is called *feature extraction* in pattern recognition (that is, forming new features out of given ones in order to obtain short descriptions). We will extend Banerji's framework in a different direction by concentrating on **background knowledge**: interdependencies between given features or *properties* (section 7). Also, we will investigate the algebraic properties of the model (section 6).

We will start with the definition of concepts. A *concept* is a set of objects. Philosophically speaking, this set represents the extensional meaning of a concept [11]. Here, an *object* is a member of

some given *universe of discourse*, or shortly *universe*, which we shall denote as U. (The term object will be redefined in a slightly different way). A concept is thus a subset of the universe. We will not allow arbitrary subsets of the universe as concepts, but instead assume a given set of *basic concepts*. New concepts can be formed out of the basic concepts by means of intersection, union and complementation. Furthermore, we assume that these basic concepts are grouped together to form partitions on the universe. These (given) partitions are called *properties*, and the basic concepts are called *values* (of the corresponding property). Thus, 'colour' can be viewed as a partition of a universe containing objects with exactly one colour; the value 'red' of the property 'colour' is that equivalence class of the universe, which contains precisely the red objects. Note, that the interpretation of basic concepts as values of properties includes the case of arbitrary basic concepts, which can be combined with their complements to form partitions. Note also, that in database literature a property is called an attribute. These ideas are formalised in the following definitions.

DEFINITION 1.
An *environment* is an ordered pair $<U, PP>$, where U is a non-empty set, and PP is a finite family of non-trivial partitions on U. (Non-trivial here means: the partition has at least two elements, and each element of the partition is non-empty.) U is called the *universe* of the environment; each element of PP is called a *property* in the environment. If P is a property then each element p of P is called a *value* of P.
A *concept* is defined recursively as follows:
- a value of a property is a concept;
- if A and B are concepts, then $A \cup B$ is a concept;
- if A is a concept, then $\overline{A}$ is a concept;
- nothing is a concept unless it follows from the foregoing three clauses.

□

Note, that if A and B are concepts, $A \cap B$ is also a concept. The set of all concepts in an environment $<U, PP>$ is denoted as C_{PP}; the empty concept is identical to the empty set $\emptyset$. Clearly, if $<U, PP>$ is an environment and $PP' \subseteq PP$, then $<U, PP'>$ is also an environment, and $C_{PP'} \subseteq C_{PP}$. The concepts in C_{PP} that are also members of $C_{PP'}$ are said to be *generated* by the properties in PP'.

Two members a and b of the universe U are called *indiscernible*, iff for each concept $X \in C_{PP}$, $a \in X \Leftrightarrow b \in X$. In fact, this needs only be checked for each value of each property.

FACT 1.
Let $<U, PP>$ be an environment. Two elements a and b of U are indiscernible in this environment, iff for each property $P \in PP$, a and b belong to the same value of P.

□

The relation of indiscernibility on U, as defined above, is an equivalence relation, as can easily be established. Its equivalence classes are the smallest concepts that can be constructed by means of the given properties. Algebraically speaking, the partition associated with this indiscernibility-relation is the finest partition that can be constructed out of the members of PP. The members of this finest partition on U will be called *objects*. Note, that the term object now has been redefined, meaning sets of indiscernible elements of the universe rather than single elements.

Such a partition of 'building blocks' can also be used to construct an approximation space, in which an arbitrary subset of the universe can be roughly described by the largest union of building blocks it contains, and the smallest union of building blocks it is contained in. This idea of 'rough sets' has been originally proposed by Pawlak [12]. Rough sets could be the 'missing link' between algebraic and statistical learning methods [5].

From the foregoing follows, that each object is a non-empty set. Moreover, each object is equal to an intersection of one value of each property. In general this statement can not be reversed, as some of the intersections of one value of each property might be empty. Let $PP = \{P_1, \ldots, P_n\}$, and let

m_i ($1 \le i \le n$) be the number of values of property P_i, then the number of objects N_o is smaller than or equal to $\prod_{i=1}^{n} m_i$. If the number of objects is equal to this product, the environment is called *full*. Consequently, in a full environment, each intersection of one value of each property is an object. Note, that the number of concepts is finite as long as the number of properties and hence the number of objects is finite, as was assumed in the preceding definition. To be precise, the number of concepts equals 2^{N_o}.

The question whether an environment is full can be answered if one has exact knowledge about the elements of the universe each value of each property contains. However, as the shift from elements of the universe towards objects as smallest concepts indicates, we are not at all interested in individual elements of the universe, but only in concepts†. Consequently, it can only be determinated whether an environment is full, if it is known which intersections of one value of each property are empty. This, or rather an efficient encoding thereof, is what is called 'background knowledge' in this paper (see section 7).

It will prove fruitful to define some special classes of concepts. One of them, the class of objects, has already been encountered, and it has been shown that each object is equal to an intersection of values, one of each property. A concept that is equal to the intersection of some values will be called a *conjunctive concept*. Conjunctive concepts have widely been recognised as being relatively easy to learn [13, 14, 15]. Another important class of concepts is the class of *simple concepts*. A simple concept is an intersection of terms, each term being a union of values of the same property. This type of concept has received some attention ([15, 16], where it is called internal disjunction). For the formal definition of conjunctive and simple concepts, two functions CG and SG are defined, yielding the smallest conjunctive resp. simple superset of an arbitrary subset of the universe. Conjunctive resp. simple concepts are then defined as the fixpoints of these functions. Recursive definitions could have been used instead, but these two functions will be used in the algebraic discussions in section 6.

DEFINITION 2.
Let $<U, PP>$ be an environment, $PP = \{P_1, \ldots, P_n\}$, $P_i = \{p_{i1}, \ldots, p_{im_i}\}$, $1 \le i \le n$. Let $CG: \Pi(U) \rightarrow C_{PP}$ (where $\Pi(U)$ denotes the powerset of U) be defined as

$$CG(X) = \bigcap \{p_{ij} \mid X \subseteq p_{ij}\}$$

$CG(X)$ will be called the *conjunctive generalisation* of X and denoted as X^C. Clearly $X \subseteq X^C$ for every $X \subseteq U$. A concept $Y \in C_{PP}$ is a *conjunctive concept* iff $Y^C = Y$.

Let $SG: \Pi(U) \rightarrow C_{PP}$ be defined as

$$SG(X) = \bigcap_{i=1}^{n} \bigcup_j \{p_{ij} \mid p_{ij} \cap X \neq \emptyset\}$$

$SG(X)$ will be called the *simple generalisation* of X and denoted as X^S. Clearly $X \subseteq X^S$ for every $X \subseteq U$. A concept $Y \in C_{PP}$ is a *simple concept* iff $Y^S = Y$.

□

FACT 2.
For all $X \subseteq U$, $X \subseteq X^S \subseteq X^C$.
Moreover, all conjunctive concepts are simple.

□

Again, there are interesting parallels with Pawlaks approximation space, subsets of the universe this time being approximated by their simple and conjunctive generalisation, respectively.

† Note that, once this is said, universa are no longer needed. Instead, we could define an abstract algebra of concepts, generated by values of properties. However, this set-theoretical model is retained because of its appeal to intuition and its links to Predicate Logic. After all, the algebra of concepts as sets is isomorphic to such an abstract algebra.

EXAMPLE 1.
Consider a universe of discourse U, consisting of two-dimensional figures: large triangles, medium-size triangles, small triangles, small squares, and small parallelograms. Each of these categories contains objects of equal size, of four different colours: green, yellow, red, and blue. As these figures are precisely the figures that are contained in a tangram set (ignoring colour), we call our little universe a 'tangram universe'; see figure 1.

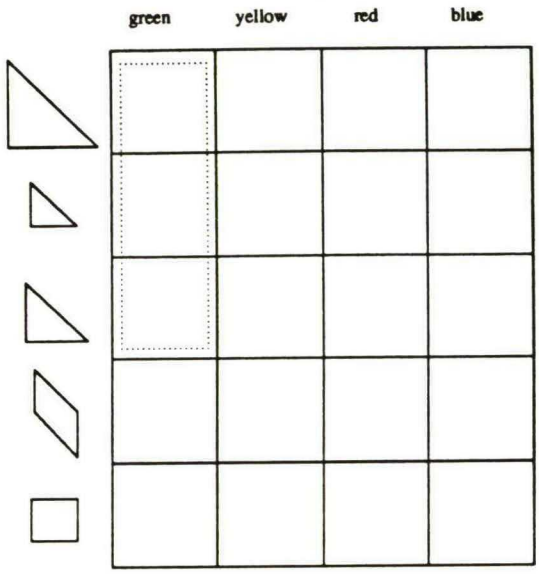

Figure 1. A tangram universe

If we define two properties 'colour' and 'shape' on U, with the values mentioned above, we can interpret figure 1 also as representing the environment $<U$, {colour, shape}$>$. The values of 'colour' are the four vertical bars, and the values of 'shape' are the five horizontal bars. Each square in the figure, obtained by intersecting a horizontal and a vertical bar, represents an object (e.g., 'red squares'), the elements of which are indiscernible.
It follows from the specification above, that this environment is full. However, we might have specified instead, that red squares do not exist in our universe. This is an example of background knowledge, which has to be represented in the picture of the environment in some form or another.
The concept of 'green triangles' is indicated in the figure. This concept is simple: 'green objects that are either small triangles, medium size triangles or big triangles'; but it is not a conjunctive concept. Its conjunctive generalisation is 'green objects'.

□

Now, the outline of the set-theoretical model is complete. In the next section, a short survey is given on how this model can be combined with a description language. After that, the use of the model in concept learning from examples is discussed.

4. Description languages

In the preceding section, a precise mathematical meaning has been given to such notions as property, concept and object. Still, a learning system does not process concepts, but rather descriptions of concepts. As has been argued elsewhere [17], the distinction between a concept and its description is crucial. Descriptions of concepts are expressed in a *description language*, for which the set-theoretical model serves as a semantics. Indeed, the model looks much like a Tarskian interpretation for first order

Predicate Logic [18], the main difference being the introduction of properties grouping together basic concepts, and the disregard for individual elements of the universe. Any suitable description language should include a device for expressing properties and corresponding values. These remarks are not meant to intend that first order Predicate Logic is not a suitable description language: on the contrary, when multiple environments are considered, the introduction of variables as in Predicate Logic becomes necessary, as will be shown in section 8.

In this section, we describe a simple description language for use in the rest of the paper. Some parts of the language are indicated below, but will be elaborated in following sections. The language, which is of course not context-free (values of properties need to be declared, for instance), is specified by a context-free grammar combined with informally specified context conditions.

DEFINITION 3.

<learning task>	→	*<environment decl>* *<concept decls>* *<examples>* .
<environment decl>	→	*<environment name>* = *<property decl>* $^+$ *<background knowl>* *
<property decl>	→	*<property name>* : *<value name>* (, *<value name>*)$^+$;
<concept decls>	→	(*<concept name>* = *<concept expr>* ;)*
<concept expr>	→	*<property name>* IS *<value name>*
		| *<concept name>*
		| (*<concept expr>* AND *<concept expr>*)
		| (*<concept expr>* OR *<concept expr>*)
		| NOT (*<concept expr>*)

Context conditions:

- within an environment declaration, each property name should be unique;
- within a property declaration, each value name should be unique;
- within a concept expression, each pair (property name, value name) should occur in a property declaration;
- concept names should be declared in a non-circular fashion.

Some special classes of concept expressions can be defined, which are related to the previously defined classes of objects, simple concepts and conjunctive concepts. The terminal symbols AND, OR and NOT are called *connectives*, (is called *left brace* and) is called *right brace*. The construction *<property name>* IS *<value name>* is called a *descriptor*. A *conjunctive expression* is a concept expression consisting of descriptors, braces and connectives AND. A conjunctive expression containing descriptors with different property names is called a *proper* conjunctive expression. A *disjunctive expression* is a concept expression consisting of descriptors, braces and connectives OR. An *object expression* is a (proper) conjunctive expression, where each property name is contained in a descriptor exactly once. A *simple expression* is a concept expression consisting of simple disjunctions, braces and connectives AND, where a *simple disjunction* is a disjunctive expression containing the same property name in each descriptor.

□

The non-terminal symbols *<examples>* and *<background knowl>* will be specified in later sections.

The semantics of this concise description language is, at first sight, obvious and straightforward. Given an environment declaration, let *PVN* denote the set of pairs *<pn, vn>* such that *vn* is a value name occurring in the property declaration of the property name *pn*. Furthermore, let *<U, PP>* be an environment, and let *VV* denote the set of values in this environment (thus $VV = \bigcup\{P \in PP\}$). An *interpretation* is a function I: $PVN \rightarrow VV$, such that $I(pn, vn_1) = p_1$ and $I(pn, vn_2) = p_2$ imply that p_1 and p_2 are values of the same property. *<U, PP>* is a *valid environment* for the environment

declaration iff I is bijective. In other words, an interpretation relates property names to properties, and value names to values, such that the environment declaration correctly describes which value belongs to which property. This mapping is easily extended to a mapping of concept expressions to concepts, such that each concept expression is assigned a unique concept, the *meaning* of the expression with respect to the interpretation. This extended mapping will not be described in detail. Note, that while it has been established that the set of concepts C_{PP} is finite, the set of concept expressions is infinite. However, the equality relation '=' between concepts can be used to define an equivalence relation between concept expressions: two concept expressions are *equivalent* (with respect to an interpretation) iff their meanings (with respect to that interpretation) are equal. From this point on, the phrase 'with respect to an interpretation' will be omitted if no confusion can possibly arise.

The meaning of an object expression with respect to an interpretation is either an object, or it is the empty concept. A *maximally* valid environment is an environment, such that the meaning of an object expression is an object. Consequently, each maximally valid environment is full. In order to describe non-full environments correctly, background knowledge has to be supplied, reducing the set of object expressions. This will be elaborated in section 7. In the sequel, we will only consider maximally valid environments, omitting the adverb 'maximally'.

EXAMPLE 2.
Consider the environment of figure 1. It is a valid environment for the following environment declaration:

```
tangram =   shape :   large-triangle , medium-triangle ,
                      small-triangle , square ,
                      parallelogram ;
            colour :  green , yellow , red , blue ;
```

We could introduce a name for the indicated concept as follows:

```
green-triangles =  ( ( ( shape IS large-triangle OR shape
                   IS medium-triangle ) OR shape IS
                   small-triangle ) AND colour IS green )
```

The concept expression in this concept declaration consists of two simple disjunctions connected by AND, and hence is a simple expression. It can be easily shown, that under the intended interpration, the indicated concept in figure 1 is indeed the meaning of the above concept expression. By the same token, it can be shown that the following concept expression is equivalent with it:

```
NOT ( ( NOT ( colour IS green ) OR ( shape IS parallelogram
OR shape IS square ) ) )
```

The tangram environment is a maximally valid environment for the above environment declaration. If red squares didn't exist, the environment would still be valid, although not maximally valid. In order to describe it fully, we have to introduce in our description language a way to describe such a phenomenon. We will return to this issue in section 7.

□

5. Learning

As has been stated in section 2, verbal concept learning from examples requires a teacher supplying example-descriptions of example-objects to a learner (the word object is used here in its intuitive meaning, namely a member of the universe). If the teacher uses a description language as described in the previous section, an example-description is necessarily a concept expression representing a concept (a subset of the universe). Of course, this concept contains the example-object, but it may contain other elements of the universe. Now, how is such an example-description to be interpreted?

In this paper, the view is advocated that the teacher is not always in the position to state that what he knows about the example-object (i.e., whether it belongs to the concept to be learned) is equally true of each element of the concept which is the meaning of the example-description. Consequently, a positive example-description should be interpreted as: there exists an element of the universe

which is both a member of the concept to be learned L and the meaning C of the example-description, thus $L \cap C \neq \emptyset$. Similarly, if C' is the meaning of a negative example-description, the only conclusion the learner can safely draw is $\overline{L} \cap C' \neq \emptyset$.

It is certainly true that adopting such a cautious approach towards interpreting the information supplied by the teacher does not result in improving the convergence of any learning algorithm towards a unique solution of the learning problem. Yet, we believe that, in general, teachers supply incomplete knowledge with which learning systems should be able to deal, not by 'jumping to conclusions', but by being relatively robust. To be more specific: a learning algorithm should, in our view, be built upon four foundations:

- a clear understanding of what it means to say that the output of the algorithm is consistent with its input;
- insight in the complexity of the task, in terms of the mathematical properties of both possible input and desired output;
- a motivated choice of the control mechanism (for example, whether the order of the given examples will be considered significant; whether a set of hypotheses will be reduced or rather one hypothesis will be subsequently refined);
- a motivated choice of the heuristic rules, applied for improving convergence towards one plausible solution.

The rest of this paper deals with the first two points exclusively.

Regarding the first point above, which comprises the precise specification of the learning task, our standpoint has already been indicated. It now can be reformulated as follows: the rule that 'what the teacher conveys about the example-object he has in mind is equally true of everything else that satisfies the example-description' is a (perhaps very useful) heuristic rather than a deep insight about the relation between example-descriptions and the environment, and should be treated accordingly. Concerning the second point mentioned above, algebraic analysis of the set of concepts as described in the next section will reveal properties of conjunctive and simple concepts which can be successfully exploited when devising a learning algorithm.

We proceed by specifying the grammar rule for examples.

DEFINITION 4.

<examples>	→	*<example desc>* $^+$
<example desc>	→	POSITIVE : *<concept expr>*
		| NEGATIVE : *<concept expr>*

□

Our next task is, to specify consistency conditions for concept expressions with respect to a set of example-descriptions. We prefer to formulate these constraints in the domain of concepts rather than in the domain of concept expressions. In the sequel, we assume to have at our disposal an environment declaration, a maximally valid environment and an interpretation, such that the environment is indeed fully described by the environment declarations. Consequently, we can assign to each example description an *example*, being the meaning of the concept expression in the example description. An example is *positive* iff its example-description is, and *negative* otherwise.

DEFINITION 5.

Given an environment $<U, PP>$, a concept C is *consistent with example* e iff e is a positive example and $e \cap C \neq \emptyset$ or e is a negative example and $e \cap \overline{C} \neq \emptyset$. Let there be specified a set of positive examples PE and a set of negative examples NE, a concept C is *consistent with the examples* iff C is consistent with each member of PE and NE. The set of concepts that is consistent with the specified examples is referred to as the *consistency set*.

□

It is possible, that no concept is consistent with the examples. This is, in general, a symptom of a defect in the examples supplied by the teacher. Of course, the teacher may have made a mistake. However, it is possible that the symptom is caused by the circumstance, that the environment the teacher uses for his examples does not contain the concept to be learned as a concept. Note, that this circumstance can possibly be detected by the learner, but never be remedied.

As objects are the building blocks of concepts, the formula $e \cap C \neq \emptyset$ above implies that there is an object o that is contained in $e \cap C$. An interesting case arises, when e itself is an object, because then we can conclude that e is completely contained in the concept to be learned. An example which is an object will be called a *complete* example. Any concept which is consistent with the examples, should contain each positive complete example, and its complement should contain each negative complete example. Complete examples convey a maximum of information to the learner, and it is only in this case that we can conclude that anything the teacher knows about the example-object he has in mind is equally true of everything else that satisfies the example-description.

EXAMPLE 3.
Consider the environment declaration of the previous example. The following are example descriptions.

POSITIVE : (colour IS yellow AND shape IS small-triangle)
NEGATIVE : colour IS red

The examples associated with these example descriptions, *A* and *B* respectively, are depicted in figure 2.

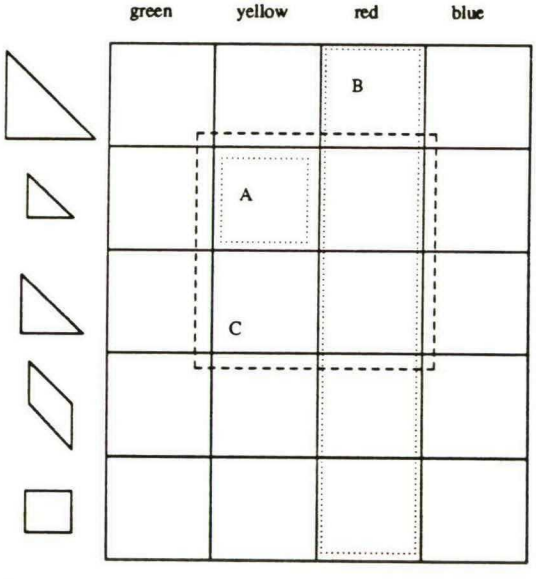

Figure 2. Examples in the tangram environment

As is immediately clear from this picture, *A* is a complete example and *B* is not. Concept *C* is consistent with both examples. *C* is not consistent with *A* if the latter concept would be a negative example; *C* would still be consistent with *B* if it were a positive example. In fact, *B* could be both a negative example and a positive example in the same learning task, without causing the consistency set to be empty. Clearly, this is not valid for any complete example.

□

6. Algebraic theory of learning

The algebraic theory of concept learning from examples as presented here is based on some rather obvious observations.

FACT 3.
Let $<U, PP>$ be an environment, and let C_{PP} denote its set of concepts. The algebra $< C_{PP}, \cap, \cup, \bar{}, \emptyset, U >$ (where $\cap$, $\cup$ and $\bar{}$ denote set-theoretical intersection, union and complement relative to U respectively) is a Boolean algebra, partially ordered by set-inclusion, and is called the *concept algebra*. The concept algebra is finite; its atoms are the objects of the environment. Let O_{PP} denote the set of objects, then the Boolean algebra $< \Pi(O_{PP}), \cap, \cup, \bar{}, \emptyset, O_{PP} >$ (where $\Pi(O_{PP})$ denotes the powerset of O_{PP}, and $\bar{}$ is taken relative to O_{PP}) is isomorphic to the concept algebra, and is called the *object algebra*. Both algebras will often be denoted by their carrier sets C_{PP} and $\Pi(O_{PP})$. The isomorphism from C_{PP} to $\Pi(O_{PP})$ is denoted as *obj*, thus $obj(C)$ is the set of objects that are contained in C.

□

At any time, we can use the algebra that is most convenient for our purposes. What, now, are the algebraic properties of the consistency set (the set of concepts that are consistent with some prespecified examples)? Clearly, this set is a subset of C_{PP} and can be constructed by taking each individual example and removing the concepts that are not consistent with the example. We have discussed four types of examples (along two 'dimensions': positive/negative and complete/incomplete) and will study this reduction step for each type.

Let e be a positive complete example. Each concept that is consistent with this example contains it completely, and thus we can remove from $\Pi(O_{PP})$ all object-sets that don't contain e. The resulting set is still a Boolean algebra with $\{e\}$ as minimal element (and complement taken relative to $O_{PP}-\{e\}$). Similarly, if e' is a negative complete example, then all object-sets that do contain e' have to be removed from $\Pi(O_{PP})$, resulting in the set $\Pi(O_{PP}-\{e'\})$, which constitutes also a Boolean algebra.

The situation concerning examples that are not complete is somewhat more difficult. In particular, let e be a positive example and let $obj(e) = \{o_1, \ldots, o_k\}$, $k>1$, then only those object-sets can be removed from $\Pi(O_{PP})$ which do not contain at least one o_i, $1 \leq i \leq k$. This results in a set with k minimal elements (to wit: $\{o_1\}, \ldots, \{o_k\}$). Clearly, this is not a Boolean algebra. Similarly, if e were a negative example, the result would be a set with k maximal elements (to wit: $O_{PP}-\{o_1\}, \ldots, O_{PP}-\{o_k\}$).

While (finite) Boolean algebras can be concisely described by summing up their atoms and specifying the operations of the algebra, the consistency set lacks this quality iff some of the examples are not complete. Obviously, the complexity of the learning task increases when incomplete examples are taken into account. In any realistic learning situation, the consistency set is very large, and it is therefore not feasible to construct the entire consistency set before pruning it with the aid of heuristics. Rather, a learning algorithm should limit attention to a restricted subclass of concepts, which are not only fewer in number, but also show some pleasing algebraic properties. Moreover, such an approach enjoys a heuristic justification as well, as Wittgenstein points out [19]:

> "Der vorgang der Induktion besteht darin, dass wir das einfachste Gesetz annehmen, das mit unseren Erfahrungen in Einklang zu bringen ist."

Obvious candidates for serving as such a restricted subclass of concepts are the classes of conjunctive and simple concepts, which are the subjects of the rest of this section. A final remark concerning the consistency set: the algebraic theory developed so far is already powerful enough to give correctness proofs of algorithms. In [10] an algorithm is given for the construction of the consistency set which is demonstrated to be correct by mathematical proof.

The definition of conjunctive concepts and simple concepts has already been given in section 3. Let C^C_{PP} denote the set of conjunctive concepts, and let C^S_{PP} denote the set of simple concepts. It is easily demonstrated that these sets are closed under intersection: for instance, $X \cap Y$ is a simple concept iff both X and Y are simple concepts. C^C_{PP} and C^S_{PP} are not closed under union. However, the following statement is not difficult to prove.

FACT 4.
In any environment $<U, PP>$, the sets of conjunctive and simple concepts C^C_{PP} and C^S_{PP} are lattices under inclusion. The least upper bound of two conjunctive concepts X and Y in the lattice $< C^C_{PP}, \subseteq >$ is $(X \cup Y)^C$; the least upper bound of two simple concepts V and W in the lattice $< C^S_{PP}, \subseteq >$ is $(V \cup W)^S$.

□

While C_{PP} constitutes a Boolean algebra, $< C_{PP}, \subseteq >$ is also a lattice, with least upper bound $X \cup Y$ of two concepts X and Y. Consequently, if e_1 and e_2 are two positive complete examples, the smallest concept that is consistent with e_1 and e_2 is $e_1 \cup e_2$. If we define a (proper) *generalisation* of some concepts as a (proper) superset of the union of those concepts, we see that $e_1 \cup e_2$ is not a proper generalisation of e_1 and e_2. With conjunctive and simple concepts, things are different: $(e_1 \cup e_2)^C$ and $(e_1 \cup e_2)^S$ are in most cases proper generalisations of e_1 and e_2. A restriction to conjunctive or simple concepts does not only result in a reduction of the complexity of the learning task, but also improves the generalisation capability of a learning system (at the cost of decreased universality).

Some of these statements are rephrased in the following FACT.

FACT 5.
Let $<U, PP>$ be an environment, and let e_1 and e_2 be two complete examples (objects). The smallest conjunctive generalisation of e_1 and e_2 (that is, the smallest conjunctive concept that contains both e_1 and e_2) is equal to the intersection of those values $p \in P \in PP$, for which $e_1 \subseteq p$ and $e_2 \subseteq p$. The smallest simple generalisation of e_1 and e_2 is equal to $\bigcap_i (p_i \cup q_i)$, with $p_i \in P$, $q_i \in P$, $P \in PP$ and $e_1 \subseteq p_i$, $e_2 \subseteq q_i$ $(1 \leq i \leq n)$.

□

The first part of this fact is a mathematical paraphrase of the learning paradigm frequently found in the literature [13, 14]: ignore the differences and retain the correspondences between the examples. It appears, however, as a very specific case in the theory presented here; a case that can only be justified when dealing with complete examples and searching for conjunctive concepts.

EXAMPLE 4.
Consider the following figure.

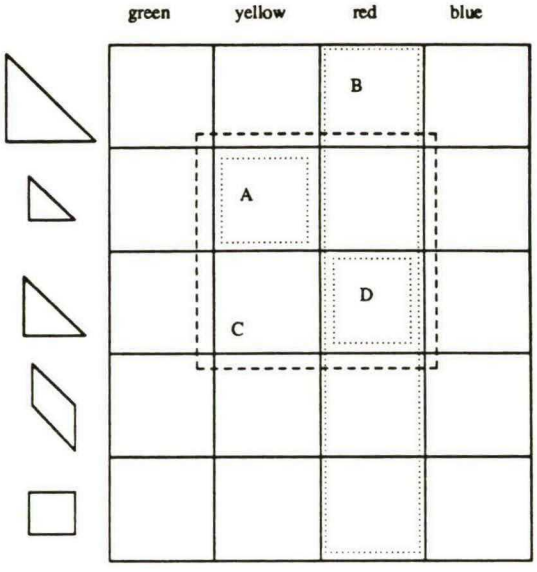

Figure 3. Another learning task in the tangram environment

The smallest conjunctive generalisation of A and D is the universe U; that is, any negative example (such as B), be it complete or not, together with the positive examples A and D would lead to the conclusion that the concept to be learned can not be conjunctive. The smallest simple generalisation of A and D is the concept C.

□

This section concludes with the specification of some complexity measures. Let $<U, PP>$ be a full environment, $PP = \{P_1, \ldots, P_n\}$, and let m_i ($m_i>1$) be the number of values of property P_i ($1 \le i \le n$). As indicated before, the number of objects in such an environment is $N_o = \prod_{i=1}^{n} m_i$, and the number of concepts is 2^{N_o}. A straightforward inductive argument shows that the number of conjunctive concepts is $N_C = 1 + \prod_{i=1}^{n}(m_i+1)$, and the number of simple concepts is $N_S = 1 + \prod_{i=1}^{n}(2^{m_i}-1)$. As an example, let there be two properties with number of values three and four, then the number of objects is 12, the number of concepts is 4096, the number of conjunctive concepts is 21 and the number of simple concepts is 106.

7. Background knowledge

As has been stated in section 4, environment declarations are intended to describe maximally valid environments, from which it follows that environment declarations describe only full environments (environments with the maximum number of objects). In order to describe non-full environments properly, environment declarations have to be augmented with *background knowledge*. In its simplest form, background knowledge states that some concepts are empty. Of course, such a statement needs only be made if it cannot be deduced otherwise from the environment declaration. For instance, the meaning of the concept expression 'colour is red and colour is blue' is empty in any valid environment.

The *primary background knowledge rule* is thus a statement that satisfies the following grammar rule.

<background knowl> → *<concept expr>* NOT EXISTS

It is easy to see, that the expressiveness of this grammar rule is essentially not decreased if *<concept expr>* is a proper conjunctive expression, where each property name is contained in a descriptor only once. Such an expression was previously called an object expression. Because the semantics of the above grammar rule is obviously, that the meaning of *<concept expr>* is the empty concept, *<concept expr>* does **not** describe an object. Therefore, we redefine the notion of object expression as follows: an *object expression* is a proper conjunctive expression, where each property name occurs in a descriptor exactly once, which does not occur in a primary background knowledge rule. The definition of a maximally valid environment remains unchanged (an environment in which the meaning of any object expression is an object); hence, it depends on the existence of primary background knowledge rules in the environment declaration, whether a maximally valid environment is full.

The specified primary background knowledge rule is sufficiently powerful to describe any non-full environment properly. Yet, there exist environments for which background knowledge can be codified in a more concise way. For instance, we could allow arbitrary concept expressions in the above grammar rule. A more extreme case can be described as follows. Consider an environment $<U, PP>$ with two properties P_i and P_j, such that P_i is a refinement of P_j. This means, that each value of P_j is equal to the union of one or more values of P_i. It is easy to show, that the set of concepts C_{PP} is not reduced by removing P_j from PP. It may be desirable, however, to retain P_j, because its removal does reduce the set of conjunctive concepts C^C_{PP}. Now, a lot of intersections of values of P_i and P_j are empty, and the specification of primary background knowledge rules would be cumbersome. It would be far more efficient to specify background knowledge in this situation in a way similar to the introduction of concept names:

<background knowl> → (*<property name>* IS *<value name>*)
EQUALS *<concept expr>*

where *<property name>* is the name for P_j, and *<concept expr>* is a simple disjunction (of which the property name denotes P_i)†. We will call such a rule a *secondary background knowledge rule*. Note, that from such a concise statement it should be deduced, which 'candidate object expressions' are not real object expressions, because their meaning is empty.

Relations between two properties can be described in a more general way. First, we define dependency between an arbitrary number of properties.

DEFINITION 6.
Let $<U, PP>$ be an environment, and let $PP' \subseteq PP$ contain at least two properties. The properties in PP' are *mutually independent* iff the environment $<U, PP'>$ is full, and *mutually dependent* otherwise.

□

Now a mapping is defined, that can be used to determine whether two properties are mutually dependent, to classify their 'degree of mutual dependency', and to express background knowledge.

DEFINITION 7.
Let $<U, PP>$ be an environment, $PP = \{P_1, \ldots, P_n\}$, $1 \le i \le n$, $1 \le j \le n$, and $P_i = \{p_{i1}, \ldots, p_{im_i}\}$. The function $\Phi_{ij}: P_i \rightarrow \Pi(P_j)$, with

$$\Phi_{ij}(p_{ik}) = \{q \in P_j \mid q \cap p_{ik} \neq \emptyset\},\ 1 \le k \le m_i$$

is called the *value mapping from P_i to P_j*. A value q of P_j for which $q \in \Phi_{ij}(p_{ik})$ is called a *possible value of P_j given value p_{ik} of P_i*.

□

† The concept expression might as well be an arbitrary expression containing several property names.

If a member of the universe is known to be contained in p_{ik}, then it is also contained in one of the values in $\Phi_{ij}(p_{ik})$, because $P_j - \Phi_{ij}(p_{ik}) = \{q \in P_j \mid q \cap p_{ik} = \emptyset\}$.

FACT 6.
Let $<U, PP>$, P_i, $P_j = \{p_{j1}, \ldots, p_{jm_j}\}$ and Φ_{ij} be as in the previous DEFINITION. The objects in the environment $<U, \{P_i, P_j\}>$ are exactly those concepts $p_{ik} \cap p_{jl}$ for which $p_{jl} \in \Phi_{ij}(p_{ik})$, $1 \leq k \leq m_i$, $1 \leq l \leq m_j$.
Furthermore, let $\Phi_{ji}: P_j \rightarrow \Pi(P_i)$ be the value mapping from P_j to P_i, then $p_{jl} \in \Phi_{ij}(p_{ik})$ iff $p_{ik} \in \Phi_{ji}(p_{jl})$, $1 \leq k \leq m_i$, $1 \leq l \leq m_j$.
P_i and P_j are mutually independent iff $\Phi_{ij}(p_{ik}) = P_j$ for each k, $1 \leq k \leq m_i$.

□

Thus, the value mapping completely specifies the part of background knowledge that is caused by mutual dependency of two properties. Its syntactical counterpart is therefore a useful extension of the description language.

DEFINITION 8.
A *tertiary background knowledge rule* is a statement satisfying the following grammar rule:
<background knowl> → (*<property name>* IS *<value name>*)
IMPLIES *<concept expr>*
where *<concept expr>* is a simple disjunction whose property name is not *<property name>* .

□

This tertiary background knowledge rule is intended for use in case of a 'strong' mutual dependency between two properties. It is certainly not advisable to specify such a statement for every pair of properties. For this purpose, a classification of mutual dependency is needed; a possible classification is listed below.

DEFINITION 9.
Let $<U, PP>$, P_i, P_j and $\Phi_{ij}: P_i \rightarrow \Pi(P_j)$ be as in the previous FACT.
P_j *is completely dependent on* P_i iff $\Phi_{ij}(p_{ik}) = \{p_{jl_k}\}$ for each k, $1 \leq k \leq m_i$. In this case, P_i is called a *finer property* than P_j.
P_j *is strongly dependent on* P_i iff P_j is not completely dependent on P_i, and $\Phi_{ij}(p_{ik}) \subset P_j$ (proper subset) for each k, $1 \leq k \leq m_i$.
P_j *is weakly dependent on* P_i iff P_i and P_j are mutually dependent, while P_j is completely nor strongly dependent on P_i.

□

It is easy to see, that P_i is a finer property than P_j iff P_i is a finer partition than P_j‡. Note, that the relations 'strongly dependent on' and 'weakly dependent on' are **not** symmetric.

This classification could be used as follows. If P_j is completely dependent on P_i, the value mapping Φ_{ji} is specified by means of tertiary background knowledge rules (which are, in this case only, equivalent with secondary background knowledge rules, see example below). If P_j is strongly dependent on P_i, the value mapping Φ_{ij} is also specified by means of tertiary background knowledge rules. If each value of a particular property can be constructed out of values of a set of other properties, then a secondary background knowledge rule can be used (with a concept expression containing several property names). Finally, in all other cases primary background knowledge rules should be used.

‡ Thus, the relation 'is completely dependent on' is a partial ordering on the set of properties that can be constructed out of *PP*.

A final remark: note that with each subsequent introduction of grammar rules for background knowledge, the definition of object expressions should have been modified in a straightforward way.

EXAMPLE 5.
Consider figure 4.

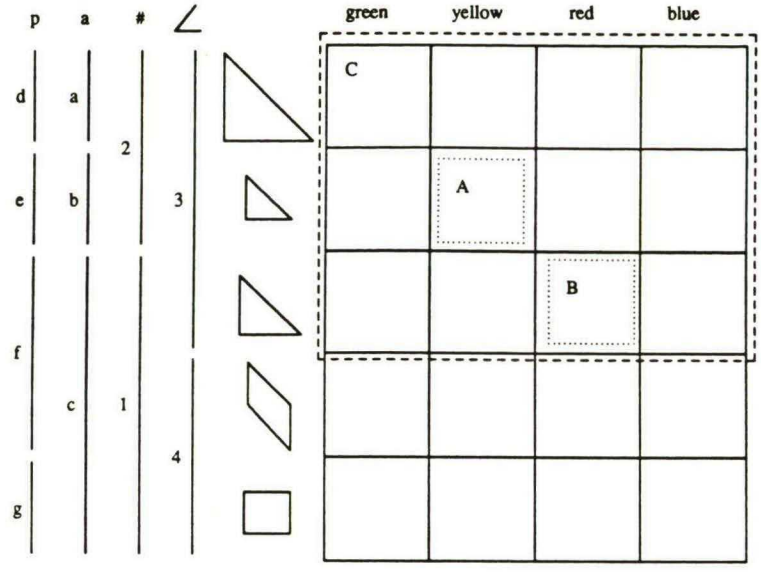

Figure 4. The extended tangram environment

In this figure, the tangram environment used in the previous examples has been extended with several properties. As these properties are strongly interrelated, the picture remains 'two-dimensional'. The environment declaration for the tangram environment is extended as follows.

```
tangram =   shape :     large-triangle , medium-triangle ,
                        small-triangle , square ,
                        parallelogram ;
            colour :    green , yellow , red , blue ;
            number :    1 , 2 ;
            angles :    3 , 4 ;
            area :      a , b , c ;
            peri :      d , e , f , g ;
```

The property 'angles' denotes the number of angles of the object; the property 'number' denotes the number of pieces of this shape and size contained in a tangram game. The properties 'area' and 'peri' denote the area and perimeter of the object.
'area' and 'peri' are finer properties than 'number', and conversely 'number' is completely dependent on both 'area' and 'peri'. As can be immediately deduced from the fact that no new concepts or objects are introduced, 'angles', 'number', 'area' and 'peri' are all completely dependent on 'shape'. 'angles' is weakly dependent on 'peri' (there is a value of 'peri', 'f', which has a non-empty intersection with all values of 'angles'), while 'peri' is strongly dependent on 'angles'.
The dependency between 'number' and 'shape' could be described by the following tertiary background knowledge rules:

```
( number IS 2 ) IMPLIES     ( shape IS large-triangle OR
                            shape IS small-triangle )
( number IS 1 ) IMPLIES     ( ( shape IS medium-triangle OR
                            shape IS parallelogram ) OR
                            shape IS square )
```

Similarly, the dependency between 'angles' and 'peri' can be described as follows.

(angles IS 3) IMPLIES ((peri IS d OR peri IS e) OR peri IS f)

(angles IS 4) IMPLIES (peri IS f OR peri IS g)

Note, that the right hand side of the latter two rules both contain the descriptor 'peri IS f', which is typical of a non-complete dependency. Alternatively, the right hand sides of the first two rules describe concepts which are disjoint, while their union covers the universe. Therefore, they may be turned into secondary background knowledge rules by changing IMPLIES into EQUALS ; this is typical of a complete dependency.

Although the introduction of these four properties does not lead to new concepts, it does lead to new concept descriptions, and to new conjunctive and simple concepts. For instance, the smallest conjunctive generalisation of *A* and *B* is *C*, described by 'angles IS 3'. (In words, the smallest conjunctive generalisation of 'yellow small triangle' and 'red medium-sized triangle' is 'triangle'.)

□

8. Learning with Predicate Logic

In section 4 a description language was introduced, of which the syntactic constructs were clearly based on the set-theoretical model of environments, properties and concepts developed in section 3. As this model contains only unary relations (i.e., values), there was no need for the introduction of individual variable symbols. In this section, we describe the use of first order Predicate Logic as a description language. This introduces two interesting topics: the use of the proof-theoretical framework of Predicate Logic (the derivation of theorems from axioms by means of rules of inference), and the description of multiple environments by introducing more individual variable symbols.

Consider a logical system with one individual variable symbol x, a finite number of (unary) predicate symbols $\pi_1, \ldots, \pi_k$, and no function symbols. The syntactical constructs (i.e., terms, atomic formulas, well formed formulas or wff's, and closed formulas or sentences) are supposed to be defined in the usual way by means of the logical connectives $\wedge$ (and), $\vee$ (or), $\neg$ (not), $\rightarrow$ (implies), $\equiv$ (equivalence), the universal quantor $\forall$ and the existential quantor $\exists$. In addition, an *expression* is a well formed formula in which no quantor occurs.

An *interpretation* for such a logical system is a pair $I = \langle U, h \rangle$, where U is a non-empty set (the *universe* of the interpretation) and h is a function that assigns to each predicate symbol π_i $(1 \leq i \leq k)$ a subset of U. The semantics of the logical system (e.g., statements like 'wff ζ is true for an interpretation I', notation $I \models \zeta$, 'wff ξ logically follows from a set Σ of wff's', notation $\Sigma \models \zeta$, and 'wff ψ is logically valid', notation $\models \psi$) is supposed to be defined in the usual way. In addition, the *meaning* of an expression ε with respect to an interpretation $\langle U, h \rangle$ is the set of elements of U that satisfy ε†.

Finally, we mention the proof-theoretical constructs of the logical system (which turn it into a logical calculus): the logical axioms (e.g., $\zeta \rightarrow (\psi \rightarrow \zeta)$), the inference rules (which are Modus Ponens: ψ follows from ζ and $(\zeta \rightarrow \psi)$, and Generalisation: $\forall x(\zeta)$ follows from ζ), and the notions of derivation and proof. If there exists a derivation of ζ from a set of wff's Σ, we write $\Sigma \vdash \zeta$ (if Σ contains a single wff ψ, we write also $\psi \vdash \zeta$); if there exists a proof of a wff ξ (i.e., a derivation from an empty set of formulas), we write $\vdash \xi$. As we just described a proper subset of first order Predicate Logic, we know that our logical system is *complete*, that is: for any set of wff's Σ and any wff ζ, $\Sigma \models \zeta$ iff $\Sigma \vdash \zeta$ (see for instance [18] pp. 62-68).

This logical system is fit for any universe U and any set of k subsets of U. Naturally, in this paper we want to restrict ourselves to the description of environments, and hence so called *non-logical axioms* have to be added in order to describe the grouping of values into properties.

---

† An element a of U *satisfies* an atomic formula $\pi(x)$, iff $a \in h(\pi)$. a satisfies a well formed formula $\pi_i(x) \wedge \pi_j(x)$ iff a satisfies $\pi_i(x)$ and a satisfies $\pi_j(x)$ (and thus $a \in h(\pi_i) \cap h(\pi_j)$). This definition extends in a similar way over the other connectives.

DEFINITION 10.
An *environment calculus* $\mathcal{EC}$ is a logical system with predicates π_{ij}, $1 \le i \le n$, $1 \le j \le m_i$, with additional non-logical axioms:

$\forall x(\neg(\pi_{ij}(x) \wedge \pi_{ik}(x)))$ for each i, j, k with $1 \le i \le n$, $1 \le j,k \le m_i$, $j \neq k$

$\forall x(\pi_{i1}(x) \vee \pi_{i2}(x) \vee \cdots \vee \pi_{im_i}(x))$ for each i with $1 \le i \le n$

$\exists x(\pi_{ij}(x))$ for each i, j with $1 \le i \le n$, $1 \le j \le m_i$

A *theorem of the environment calculus* $\mathcal{EC}$ is a wff ζ that is derivable from the logical and non-logical axioms of $\mathcal{EC}$, notation $\vdash_{\mathcal{EC}} \zeta$.

□

A non-logical axiom of the first kind expresses that two values of one property are disjoint; an axiom of the second kind expresses that the values of one property together cover the entire universe; an axiom of the third kind expresses that a value of a property is non-empty. Note, that in an environment calculus each predicate has to be distinct, while in an environment declaration a value name only needs to be unique within its property declaration.

FACT 7.
Given an environment calculus $\mathcal{EC}$, an environment $<U, PP>$, $PP = \{P_1, \ldots, P_n\}$, $P_i = \{p_{i1}, \ldots, p_{im_i}\}$, $1 \le i \le n$, and an interpretation $I = <U, h>$ such that $h(\pi_{ij}) = p_{ij}$, $1 \le i \le n$, $1 \le j \le m_i$. If a wff ζ is a theorem of $\mathcal{EC}$, it is true for the interpretation I, that is,

$\vdash_{\mathcal{EC}} \zeta \Rightarrow I \models \zeta$.

□

Note carefully, that this FACT takes the form of an implication. In order to achieve completeness, non-logical axioms have to be added that specify background knowledge. These non-logical axioms should be such, that for each expression $\pi_{1j_1}(x) \wedge \pi_{2j_2}(x) \wedge \cdots \wedge \pi_{nj_n}(x)$, $1 \le j \le m_i$, $1 \le i \le n$ (abbreviated as ζ_j) either $\exists x(\zeta_j)$ or $\neg \exists x(\zeta_j)$ is derivable from the non-logical axioms. In analogy with the previous section, possible forms for these *background axioms* could be:

just one of $\exists x(\zeta_j)$ or $\neg \exists x(\zeta_j)$;
$\neg \exists x(\varepsilon)$ for an arbitrary expression ε
(cf. primary background knowledge rule);
$\forall x(\pi_{ij}(x) \equiv \varepsilon)$
(cf. secondary background knowledge rule);
$\forall x(\pi_{ik}(x) \rightarrow (\pi_{jl_1}(x) \vee \cdots \vee \pi_{jl_j}(x))$
(cf. tertiary background knowledge rule).

If such a complete set of background axioms is added to an environment calculus, we obtain a *complete environment calculus* $\mathcal{CEC}$. It is not difficult to prove, that a complete environment calculus $\mathcal{CEC}$ is indeed complete with respect to a suitable interpretation I (thus $\vdash_{\mathcal{CEC}} \zeta \Leftrightarrow I \models \zeta$).

EXAMPLE 6.
An environment calculus for the tangram environment contains non-logical axioms like

$\forall x(\neg(\text{green}(x) \wedge \text{red}(x)))$
$\forall x(\text{green}(x) \vee \text{yellow}(x) \vee \text{red}(x) \vee \text{blue}(x))$
$\exists x(\textit{square}(x))$

A complete environment calculus for the extended tangram environment of figure 4 contains background axioms like

$\forall x(\text{number–2}(x) \equiv \text{large–triangle}(x) \vee \text{small–triangle}(x))$
$\forall x(\text{angles–3}(x) \rightarrow (\text{peri–d}(x) \vee \text{peri–e}(x) \vee \text{peri–f}(x))$

□

The main grammatical difference between an environment calculus and an environment declaration is the introduction of an individual variable symbol x. In fact, such a symbol serves no specific purpose in a description language describing only one environment. This changes if we consider a language describing so called 'multiple environments'. Until now, we have only considered unary predicates on the universe (values of properties, where a property represents a binary equivalence relation). Many realistic learning situations concern complex objects, made up from several objects, each with their own properties, with several interrelations. Structural descriptions ([2] as opposed to attribute descriptions) of complex objects require the use of several environments, each for each type of object that is part of the complex object, together with a way to combine these environments to form an environment in which the complex object can be described. The total of these environments is termed a *multiple environment* (this is not a definition, but a rather vague description). An algebraic approach to multiple environments is possible [10] but seems rather cumbersome. Predicate Logic appears to be more natural to incorporate structural descriptions, by introducing more individual variable symbols, n-ary predicate symbols, and n-ary function symbols ($n \geq 1$). An interpretation for such a calculus is a many-sorted algebra, and to each individual variable symbol a type is assigned (a sort of the algebra, or equivalently, an environment). Clearly, it is desirable to extend the learning theory presented here in this direction.

9. Learning systems

In the previous section, we have given a method to construct a logic calculus which is complete with respect to a given environment. As will be shown presently, concept learning from examples can be elegantly incorporated within that framework.

DEFINITION 11.
Given a complete environment calculus $\mathcal{CEC}$ and a symbol β which is not a predicate symbol of $\mathcal{CEC}$. A *positive example* is a well formed formula $\exists x(\zeta \wedge \beta(x))$ and a *negative example* is a well formed formula $\exists x(\psi \wedge \neg \beta(x))$, where ζ and ψ are expressions of $\mathcal{CEC}$. Let Σ_β denote the set of examples, then an expression ξ is *consistent with the examples* iff $\Sigma_\beta \nvdash_{\mathcal{CEC}} \neg (\beta(x) \equiv \xi)$; that is, it is not derivable from the examples that $\beta(x)$ is not logically equivalent with ξ.

□

We now have two methods to construct a learning system, methods which are essentially different. The first method follows from the above DEFINITION, which demonstrates that the entire theory of learning as developed in this paper can be formulated within the proof-theoretical framework of a logic calculus. Hence, a theorem prover could be used to implement a learning system; such a method could be called the *proof-theoretical approach* towards learning systems. As a complete environment calculus is complete with respect to an environment, there are no decidability problems associated with a proof-theoretical approach (the above definition might have made readers suspicious about that). The proof-theoretical approach has not yet further been investigated by the authors. Note, that direct implementation in PROLOG is not possible, because a non-logical axiom of the form $\forall x(\pi_{i1}(x) \vee \cdots \vee \pi_{im_i}(x))$, transformed to clausal form, is not a Horn-clause.

The second method for the construction of a learning system uses a description language similar to the one defined in section 4, and is based upon the view of an environment as an algebra, as described in section 6. We can term this method the *algebraic approach* towards learning systems. Alternatively, this method could be called the *semantical* approach (from a logical viewpoint). According to this method, a learning system contains the following modules:

- a **parser module** for reading in the user-defined learning task, making use of grammar rules similar to those described in this paper;
- an **algebra module** for defining the object algebra as described in section 6;
- a **semantics module** for defining the mapping of concept expressions (as defined in the parser module) onto elements of the object algebra, thus specifying the meaning of each concept

expression;

- a **learning module** for defining the notion of 'being consistent with the examples', and a further specification of the learning algorithm (that is, a specification of what is to be learned, and how it is to be learned).

A system consisting of parser module, algebra module, semantics module and a 'kernel' learning module (defining only the consistency set) can be termed a **learning shell.** Such a learning shell can be used to build a learning system by augmenting the learning module as desired. At Tilburg University, a learning shell is currently under construction, written in PROLOG.

10. Conclusion

In this paper, we have attempted to develop a sound conceptual and computational framework for learning systems. We have defined and discussed notions like concept, property, example, description language, background knowledge. In our view, the benefits of precise mathematical definitions are clear. For instance, we have shown that conjunctive and simple expressions have a special place in learning theory, not by building a learning system and concluding that the system learns them more easily than arbitrary expressions, but by proving their specific mathematical properties. Furthermore, we have argued that examples can not be identified with specific elements of the universe, but only with subsets of the universe, thus incorporating the notion of incomplete knowledge in an elegant way. A prerequisite for any learned concept is, that it is consistent with the examples supplied by the teacher. The notion of background knowledge has been discussed, revealing some 'normal forms'; however, further investigation is still required on this subject.

We have come up with two, essentially different, approaches to learning systems: the algebraic or semantical approach, and the proof-theoretical approach. It has been indicated how the notion of 'being consistent with the examples' can be implemented in a learning shell, on top of which a specific learning algorithm can be implemented, with a control strategy based on heuristics. A first version of such a shell is under way.

Topics for further investigation include: the incorporation of multiple environments within the theory, the introduction of properties with an infinite number of values, the algorithmic specification of background knowledge, and the algebra of properties in an environment. The first topic can be studied by considering a 'full' first order Predicate Logic, with several (typed) individual variable symbols, n-ary predicate symbols and n-ary function symbols. Infinite partitions can be used to model numerically measured properties such as weight. Such partitions might be inductively defined, which requires second order logic. Formal consequences of such an approach should be studied. Algorithmic specification of background knowledge means specifying the value mapping by means of an algorithm instead of argument-value pairs.

The algebra of properties in an environment consists of all properties that can be constructed out of the given ones. This does not increase the expressiveness of the environment, but influences the efficiency of concept expressions; it also increases the number of conjunctive and simple concepts. Such an algebra of properties might prove useful to formalise the notion of background knowledge further. It can also be used to study a form of learning where the learned concept contains descriptors not present in the examples, sometimes called 'constructive induction' [15]. Such a learning process has previously been mentioned by Banerji [7]; he indicates its close relationship with what is called 'feature extraction' in pattern recognition.

Acknowledgements

The authors gratefully acknowledge numerous discussions with Professor Leo Verbeek and Professor Willem Gröneveld, that have contributed considerably to the ideas discussed in this paper.

References

1. P.R. COHEN AND E.A. FEIGENBAUM (EDS.), *The handbook of artificial intelligence, Vol III*, Pitman, London, 1982.

2. T.G. DIETTERICH AND R.S. MICHALSKI, "A comparative review of selected methods for learning from examples," in *Machine Learning: an Artificial Intelligence Approach*, ed. R.S. Michalski, J.G. Carbonell and T.M. Mitchell, Springer-Verlag, Berlin, 1984.

3. R. FORSYTH AND R. RADA, *Machine Learning: applications in expert systems and information retrieval*, Ellis Horwood, Chicester, 1986.

4. G. THORNBURG AND R.S. MICHALSKI, "Machine learning: challenges of the eighties," in *Machine Learning: an Artificial Intelligence Approach, Vol. II*, ed. R.S. Michalski, J.G. Carbonell and T.M. Mitchell, Morgan Kaufmann, Los Altos, 1986.

5. S.K.M. WONG, W. ZIARKO, AND R. LI YE, "Comparison of rough-set and statistical methods in inductive learning," *International Journal of Man-Machine Studies*, vol. 24, pp. 53-72, 1986.

6. L.P.J. VEELENTURF, "An automata-theoretical approach to developing learning neural networks," *Cybernetics and Systems*, vol. 12, pp. 179-202, 1981.

7. R.B. BANERJI, *Theory of problem solving: an approach to Artificial Intelligence*, American Elsevier, New York, 1969.

8. A. BUNDY, *The computer modelling of mathematical reasoning*, Academic Press, London, 1983.

9. R. TURNER, *Logics for Artificial Intelligence*, Ellis Horwood, Chicester, 1984.

10. P.A. FLACH, *Theoretische fundering van een klasse van lerende systemen*, Twente University, Enschede, 1987. (master thesis, in Dutch)

11. W.V.O. QUINE, *From a logical point of view*, Harper & Row, New York, 1961. second revised edition

12. Z. PAWLAK, "Rough sets," *International Journal of Computer and Information Sciences*, vol. 11, pp. 341-356, 1982.

13. F. HAYES-ROTH AND J. MCDERMOTT, "An interference matching technique for inducing abstractions," *Communications of the ACM*, vol. 21, pp. 401-411, 1978.

14. C. SAMMUT AND B. COHEN, "Object recognition and concept learning with Confucius," *Pattern Recognition*, vol. 15, pp. 309-316, 1982.

15. R.S. MICHALSKI, "A theory and methodology of inductive learning," in *Machine Learning: an Artificial Intelligence Approach*, ed. R.S. Michalski, J.G. Carbonell and T.M. Mitchell, Springer-Verlag, Berlin, 1984.

16. R.S. MICHALSKI, "Pattern recognition as rule-guided inductive inference," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-2, pp. 349-361, 1980.

17. R.M. KELLER, "Defining operationality for explanation-based learning," *Artificial Intelligence*, vol. 35, pp. 227-241, 1988.

18. E. MENDELSON, *Introduction to mathematical logic*, Van Nostrand Reinhold, New York, 1964.

19. L. WITTGENSTEIN, *Tractatus Logico-Philosophicus*, Suhrkamp Verlag, Frankfurt, 1963.

ITK: P.O. BOX 90153 5000 LE TILBURG THE NETHERLANDS